Python: **Lists**

List

- A list can store a collection of data of any size.
- Python lists are one of the most versatile data types that allow us to work with multiple elements at once

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

 # a list of programming languages

['Python', 'C++', 'JavaScript']

 # list of integers

my_list = [1, 2, 3]

 # empty list

my_list = []

 # list with mixed data types

my_list = [1, "Hello", 3.4]

 # nested list

my_list = ["mouse", [8, 4, 6], ['a']]

List Is a Sequence Type

- Strings and lists are sequence types in Python
- The sequence operations for lists are the same as for strings

Operation	*Description*
`x in s`	True if element x is in sequence s.
`x not in s`	True if element x is not in sequence s.
`s1 + s2`	Concatenates two sequences s1 and s2.
`s * n, n * s`	n copies of sequence s concatenated.
`s[i]`	ith element in sequence s.
`s[i : j]`	Slice of sequence s from index i to j − 1.
`len(s)`	Length of sequence s, i.e., the number of elements in s.
`min(s)`	Smallest element in sequence s.
`max(s)`	Largest element in sequence s.
`sum(s)`	Sum of all numbers in sequence s.
`for loop`	Traverses elements from left to right in a `for` loop.
`<, <=, >, >=, ==, !=`	Compares two sequences.

Python Math

- Built-in Math Functions

```
x = min(5, 10, 25)
y = max(5, 10, 25)

print(x)
print(y)
```

```
x = abs(-7.25)

print(x)
```

```
x = pow(4, 3)

print(x)
```

The Math Module

- Python has also a built-in module called math, which extends the list of mathematical functions

import math

```
import math

x = math.sqrt(64)

print(x)
```

```
import math

x = math.ceil(1.4)
y = math.floor(1.4)

print(x) # returns 2
print(y) # returns 1
```

```
import math

x = math.pi

print(x)
```

https://www.w3schools.com/python/module_math.asp

Functions for Lists

- list1 = [2, 3, 4, 1, 32]

len(list1)

max(list1)

min(list1)

sum(list1)

- import random

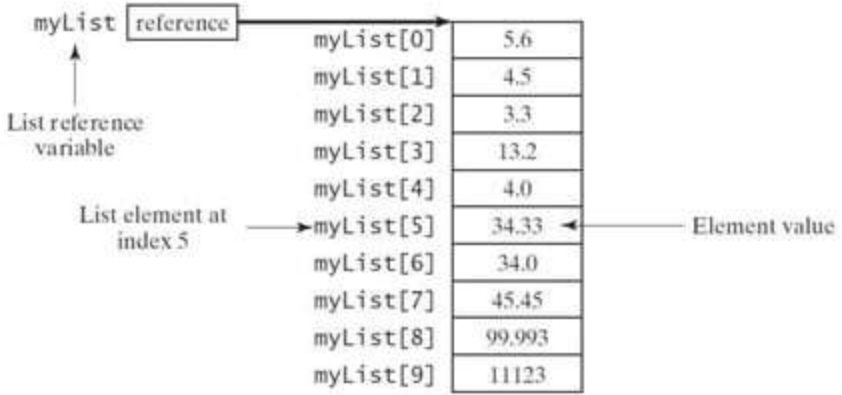
random.shuffle(list1)

Access List Elements

- Index Operator []
  - An element in a list can be accessed through the index operator, using the following syntax:

  myList[index]

  myList = [5.6, 4.5, 3.3, 13.2, 4.0, 34.33, 34.0, 45.45, 99.993, 11123]



Out of Range

- Accessing a list out of bounds is a common programming error that results in a runtime IndexError
- To avoid this error, make sure that you do not use an index beyond **len(myList) – 1**

Examples

- my_list = ['p', 'r', 'o', 'b', 'e']

- # first item
- print(my_list[0]) # p

- # third item
- print(my_list[2]) # o

- # fifth item
- print(my_list[4]) # e

```
# Nested List
n_list = ["Happy", [2, 0, 1, 5]]

# Nested indexing
print(n_list[0][1])

print(n_list[1][3])

# Error! Only integer can be used for indexing
print(my_list[4.0])
```

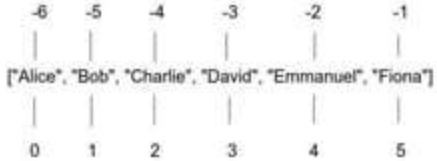
Negative indexing

- Python also allows the use of negative numbers as indexes to reference positions relative to the end of the list

list1 = [2, 3, 5, 2, 33, 21]

list1[-1]

list1[-3]



List Slicing [start : end]

- The index operator allows you to select an element at the specified index. The slicing operator returns a slice of the list using the syntax list[start : end].
- The slice is a sublist from index start to index end – 1

list1 = [2, 3, 5, 7, 9, 1]

list1[2 : 4]

list1[: 4]

list1[2 :]

list1[:]

Negative index in slicing

```
list1 = [2, 3, 5, 2, 33, 21]
 list1[1 : -3]
list1[-4 : -2]
```

The +, *, and in/not in Operators

- Concatenation operator (+) to join two lists and
- the repetition operator (*) to replicate elements in a list

```
list1 = [2, 3]
list2 = [1, 9]
list3 = list1 + list2
list4 = 3 * list1
```

The in/not in Operators

```
list1 = [2, 3, 5, 2, 33, 21]
2 in list1
```

Lexicographic order

- Lexicographical order is nothing but the dictionary order or preferably the order in which words appear in the dictionary. For example,
  - let's take three strings, "short", "shorthand" and "small". In the dictionary, **"short" comes before "shorthand" and "shorthand" comes before "small".** This is lexicographical order.

Comparing Lists

- compare lists using the comparison operators (>, >=, <, <=, ==, and !=)
  - The comparison uses lexicographical ordering:
    - the first two elements are compared, and if they differ this determines the outcome of the comparison; if they are equal, the next two elements are compared, and so on, until either list is exhausted.

```
list1 = ["green", "red", "blue"]
list2 = ["red", "blue", "green"]
list2 == list1
list2 != list1
list2 >= list1
list2 > list1
list2 < list1
list2 <= list1
```

List Comprehension: Elegant way to create Lists

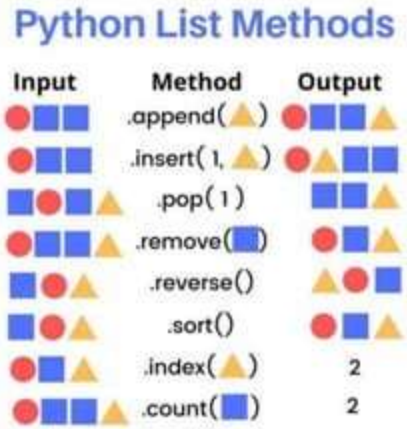
```
pow2 = [2 ** x for x in range(10)]
print(pow2)

pow2 = []
for x in range(10):
  pow2.append(2 ** x)
```

Iterating Through a List

```
for fruit in ['apple','banana','mango']:
    print("I like",fruit)
```

List Methods

Python List Methods	
Method	**Description**
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list



Examples

```
list1 = [2, 3, 4, 1, 32, 4]
list1.append(19)

list1.count(4)          # Return the count for number 4

list2 = [99, 54]
list1.extend(list2)
list1.index(4)          # Return the index of number 4

list1.insert(1, 25)     # Insert 25 at position index 1
```

More Examples

```
list1 = [2, 25, 3, 4, 1, 32, 4, 19, 99, 54]
list1.pop(2)
list1.pop()                    #returns and removes the last element from
list1
ist1.remove(32)                # Remove number 32
list1.reverse()                # Reverse the list
list1.sort()                   # Sort the list
```

List Comprehensions

- list1 = [x for x in range(5)]
- list2 = [0.5 * x for x in list1]
- list3 = [x for x in list2 if x < 1.5]

Deep and shallow copy

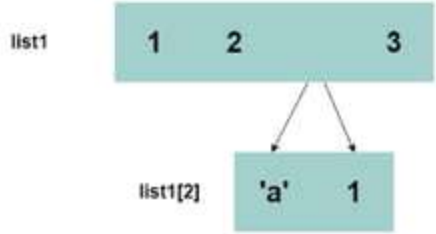
```
list1 = [1, 43]
list2 = list1
list1[0] = 22
print (list1)
print (list2)

print("----------------------------------------------")

list1 = [1, 43]
list2 = [x for x in list1]
list1[0] = 22
print (list1)
print (list2)
```

```
import copy
b = copy.deepcopy(a)
```

Nested List

- A nested list is a list of lists, or any list that has another list as an element (a sublist)
- # creating list

```
nestedList = [1, 2, ['a', 1], 3]

# indexing list: the sublist has now been accessed
subList = nestedList[2]

# access the first element inside the inner list:
element = nestedList[2][0]

print("List inside the nested list: ", subList)
print("First element of the sublist: ", element)
```



Creating a matrix

- This is done by creating a nested list that only has other lists of equal length as elements
  - The number of elements in the nested lists is equal to the number of rows of the matrix.
  - The length of the lists inside the nested list is equal to the number of columns

list1 =

	col 1	col 2	col 3	
row 1	list1[0][0]	list1[0][1]	list1[0][2]	= list1[0]
row 2	list1[1][0]	list1[1][1]	list1[1][2]	= list1[1]
row 3	list1[2][0]	list1[2][1]	list1[2][2]	= list1[2]

A matrix of size 4x3

```
# create matrix of size 4 x 3
matrix = [[0, 1, 2],
        [3, 4, 5],
        [6, 7, 8],
        [9, 10, 11]]
rows = len(matrix)  # no of rows is no of sublists i.e. len of list
cols = len(matrix[0])  # no of cols is len of sublist
# printing matrix
print("matrix: ")
for i in range(0, rows):
   print(matrix[i])
# accessing the element on row 2 and column 1 i.e. 3
print("element on row 2 and column 1: ", matrix[1][0])
# accessing the element on row 3 and column 2 i.e. 7
print("element on row 3 and column 2: ", matrix[2][1])
```

Distances list

- Each element in the distances list is another list, so distances is considered a nested list

distances = [[0, 983, 787, 714, 1375, 967, 1087], [983, 0, 214, 1102, 1505, 1723, 1842], [787, 214, 0, 888, 1549, 1548, 1627], [714, 1102, 888, 0, 661, 781, 810], [1375, 1505, 1549, 661, 0, 1426, 1187], [967, 1723, 1548, 781, 1426, 0, 239], [1087, 1842, 1627, 810, 1187, 239, 0]]

Distance Table (in miles)

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1,375	967	1,087
Boston	983	0	214	1,102	1,505	1,723	1,842
New York	787	214	0	888	1,549	1,548	1,627
Atlanta	714	1,102	888	0	661	781	810
Miami	1,375	1,505	1,549	661	0	1,426	1,187
Dallas	967	1,723	1,548	781	1,426	0	239
Houston	1,087	1,842	1,627	810	1,187	239	0

Questions

- How do you create an empty list and a list with the three integers 1, 32, and 2?
- Given lst = [30, 1, 12, 14, 10, 0],
  - how many elements are in lst?
  - What is the index of the first element in lst? What is the index of the last element in lst?
  - What is lst[2]? What is lst[-2]?
- Indicate true or false for the following statements:

  (a) Every element in a list must have the same type.

  (b) A list's size is fixed after it is created.

  (c) A list can have duplicate elements.

  (d) The elements in a list can be accessed via an index operator.

Questions

- Given lst = [30, 1, 2, 1, 0], what is the list after applying each of the following statements? Assume that each line of code is independent.
  - lst.append(40)
  - lst.insert(1, 43)
  - lst.extend([1, 43])
  - lst.remove(1)
  - lst.pop(1)
  - lst.pop()
  - lst.sort()
  - lst.reverse()
  - random.shuffle(lst)

```
squares = []
for i in range(10):
    squares.append(i**2)
print(squares)

#Another way
print([i**2 for i in range(10)])
```

Questions

- What are list1 and list2 after the following lines of code?

```
list1 = [1, 43]
list2 = list1
list1[0] = 22
```

- What are list1 and list2 after the following lines of code?

```
list1 = [1, 43]
list2 = [x for x in list1]
list1[0] = 22
```

- How do you obtain a list from a string? Suppose s1 is welcome. What is s1.split('o')?

Questions

- What is the output of the following code?

```
lst = [1, 2, 3, 4, 5, 6]
for i in range(1, 6):
    lst[i] = lst[i - 1]
print(lst)
```